QT2440 linux 安装手册

上海勤研电子

2006 年 2 月 28 日

目　录

使用指南一：如何快速为开发板安装操作系统 3
一、烧写 vivi 4
二、安装 linux 内核 6
三、安装根文件系统 7
使用指南二：如何在 LINUX 下配置开发环境 8
一、拷贝文件包 8
二、安装 ToolChain 8
三、建立映像文件夹 8
四、安装 ztelnet 9
使用指南三：如何为目标板上的 LINUX 编译程序 10
一、拷贝源代码 10
二、编译 imagewrite 10
三、编译 hello,world! 11
使用指南四：如何向目标板上的传输文件 12
使用指南五：如何使用 IMAGEWRITE 工具 14
使用指南六：如何构造自己的文件系统映像 16
使用指南七：如何编译 VIVI 18
使用指南八：如何编译内核 19
结束语 20

使用指南一：如何快速为开发板安装操作系统

本向导帮助用户给 QT2440 开发板安装 linux 操作系统，注意，原板上 NANDflash 中的系统会被破坏。与为电脑安装操作系统相比，为 QT2440 板安装 linux 是一个不同的过程，总体的步骤分为三步：

1、安装 bootloader（vivi）；

2、安装 linux 内核；

3、安装根文件系统；

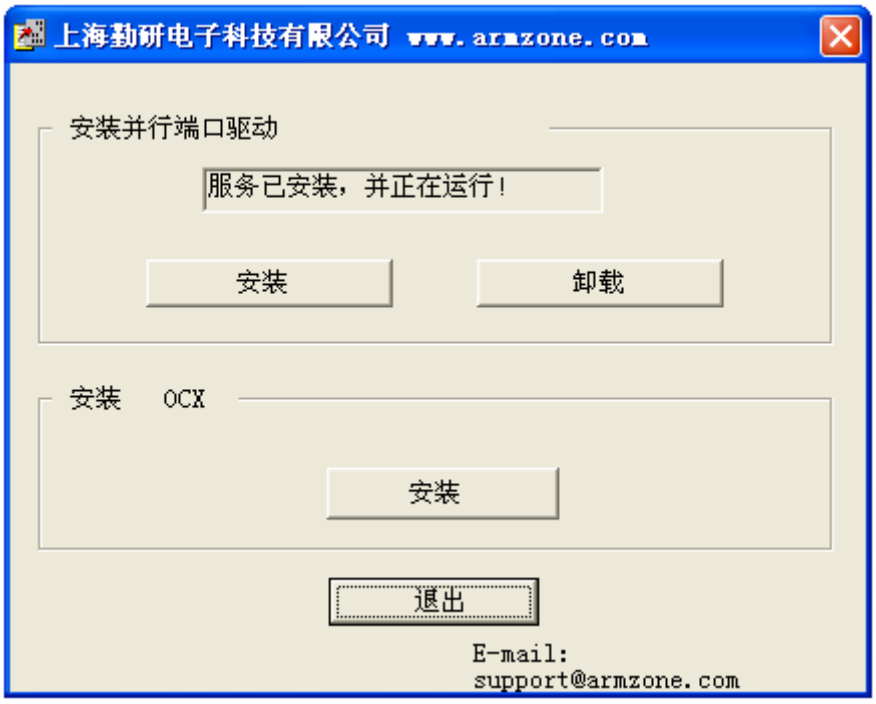
有些操作在 windows2000/XP 环境下完成，后面有些工作需要到 linux 环境下继续，推荐安装 RedHat9.0。

准备工作（安装并口 JTAG 驱动）：

将软件包解压拷贝到本地硬盘上，方便使用；随后安装 GIVEIO.SYS 驱动，windows98 不需要安装该驱动。步骤如下：

1）运行 光盘\GIVEIO 驱动安装\目录下的 armzone.exe

2）点击“安装并行端口驱动”下的“安装”按键，提示“服务已安装，并正在运行！”，表示安装成功



一、烧写 vivi

vivi：装载程序（bootloader）。就像电脑里的 BIOS 一样，vivi 起到一个引导系统的作用（vivi 是 bootloader 的名字），此外，安装系统时我们还可以用它来烧写 NANDflash；因此，需要先将 vivi 烧写到 NAND 的启动区域。我们将使用一个 JTAG 烧写工具 sjf2440.exe(光盘/image 目录下)。

重要！注意！本包中提供的工具 sjf2440.exe 不同于三星网站上的同名工具，经过我们的修改，使用套件包中 wiggle 来烧写 flash

操作流程：

1 确认 JTAG 驱动已安装。

2 确认开发板按连接要求连接(尤其是 JTAG 电缆)，加电。

3 双击运行光盘/SJF2440 目录下 vivi.bat

如下图：

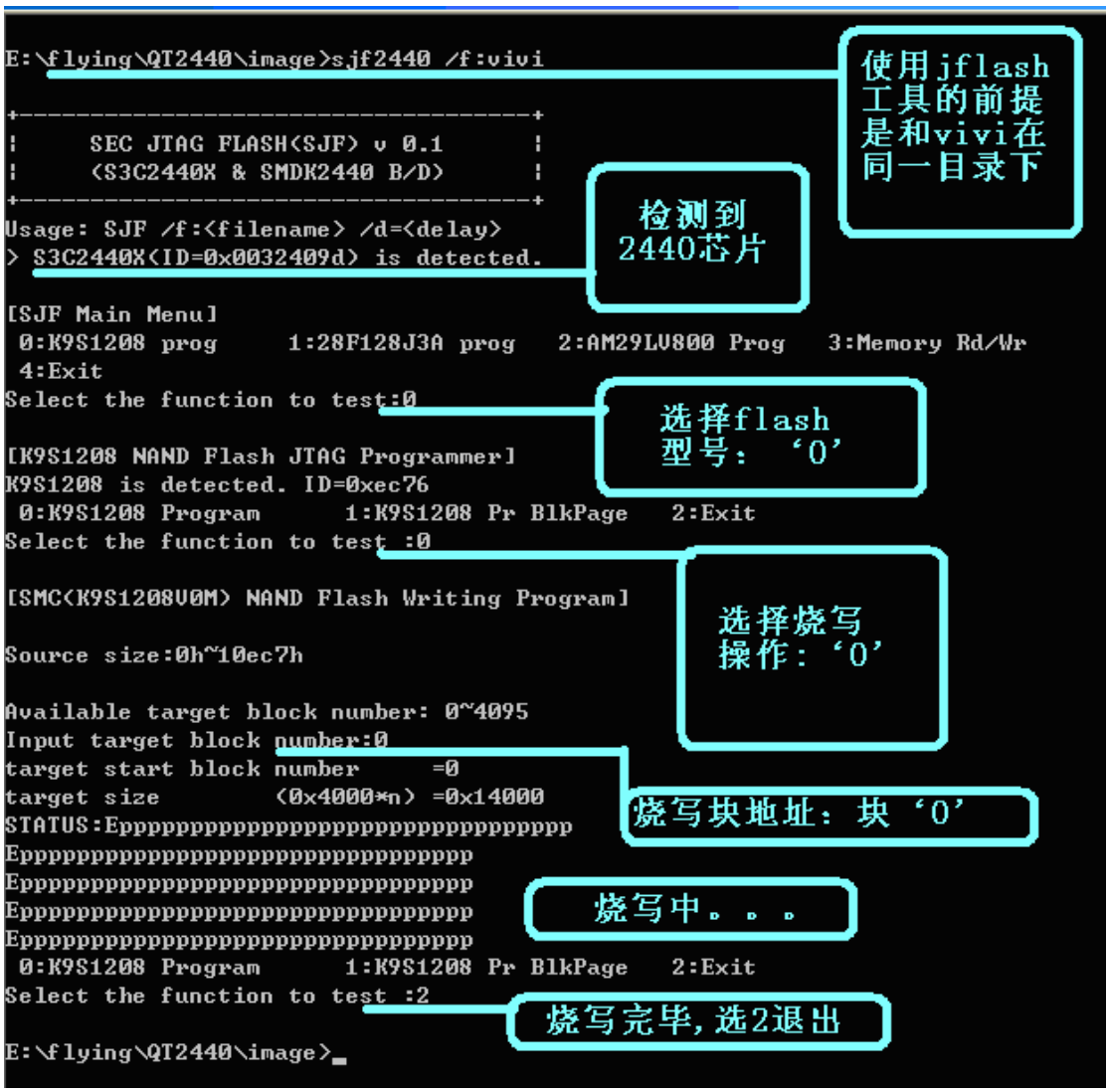


图 1 使用 sjf2440 烧写 vivi 示意

```
-+---+---++-+---+-++-+++--++-+++-+-++--+-++-+--+
```

如果烧写出现问题，请对照上图，若检测到 2440 芯片，说明 JTAG 连接无误，否则，请检查 JTAG 连接，开发板电源开关是否开启。

```
-+---+---++-+---+-++-+++--++-+++-+-++--+-++-+--+
```

烧写过程可能持续 1 分钟，烧写完毕后，打开超级终端，在配置串口属性的窗口修改：

波特率为 115200

流量控制为无。

运行超级终端后，配置 SW2（OM0:ON,OM1:ON），从 NAND 启动，开发板重新启动或重新加电，这时便可以看到串口 1 有信息输出，这便是 vivi 了；如下图（串口 1 是板子中间靠近网口编号为 P4 的 DB9 插座，P5 为串口 2，P6 为串口 3，即 COM1 COM2 COM3）

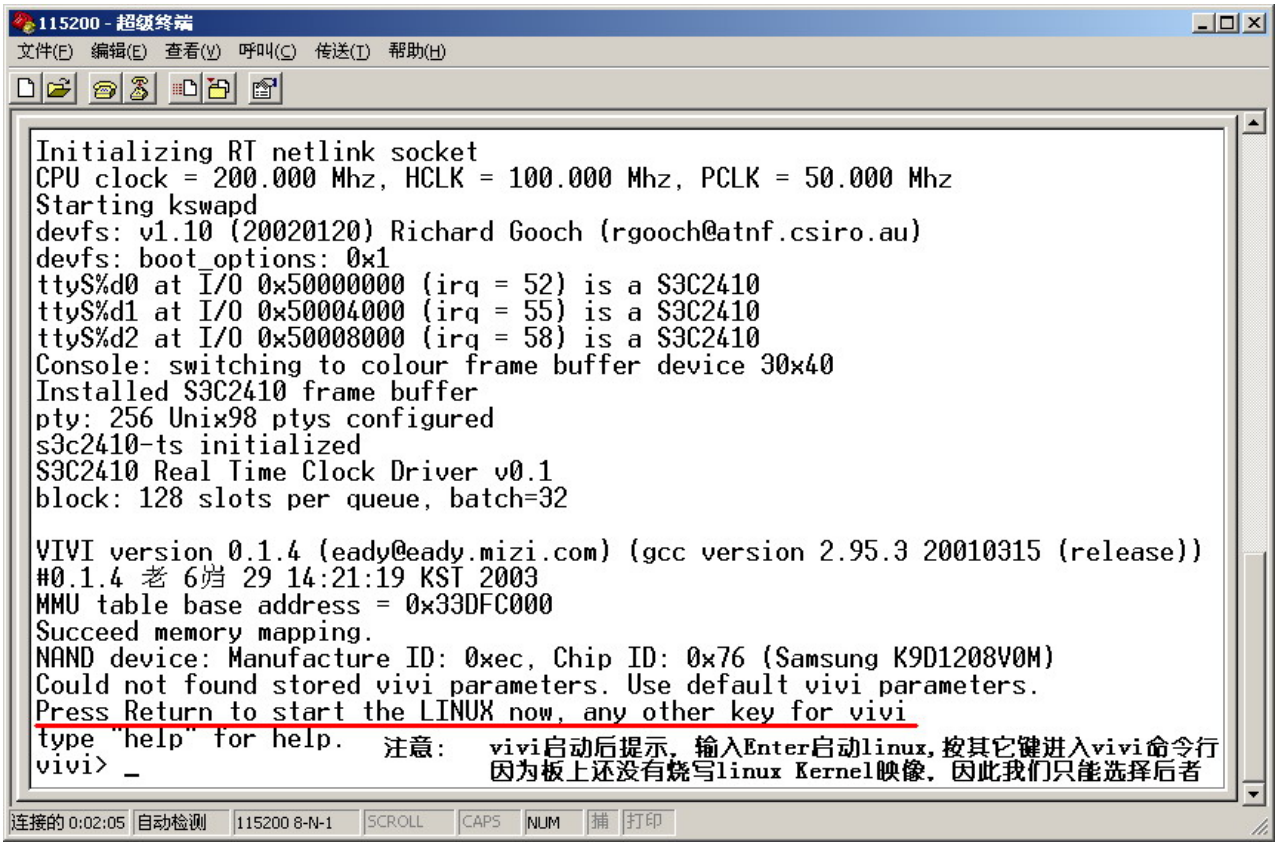


图 2 vivi 命令行界面

注意，vivi 启动后有一个按键检测，就像进入 BIOS 需要按 DEL 键一样，我们需要进入 vivi，因此，我们可以按除 ENTER 键以外其他的任何键。因为 VIVI 留给用户选择的时间很短，所以用户如果需要进入 VIVI 命令行，而不是启动 linux，请在超级终端启动后，一直按住空格键，就是一直让串口给开发板发出“0x20”，然后再启动开发板，就可以顺利进入 VIVI。

```
-+---+---++-+---+-++-+++--++-+++-+-++--+-++-+--+
```

注意：如果没有接收到任何的信息，请确认串口连接正确 PC 端的串口应该和超级终端里面端口设置一致，开发板上的串口是板子中间 P15 这个 DB9 插座。若还是没有信息请按照上面的操作重新烧写 vivi。

```
-+---+---++-+---+-++-+++--++-+++-+-++--+-++-+--+
```

二、安装 linux 内核

我们的 vivi、linux 都是烧写在 NANDflash 上运行的，在使用 NANDflash 之前，我们需要进行分区的操作。

在超级终端中，vivi 下执行

```
================================
vivi>      bon part 0 192K 2M
================================
```

分区的操作就像硬盘分区一样，最后我们的得到的结果是：

```
--------------------------------
0~192k        : vivi 驻留区域
192k~2m       : 内核驻留区域
2m~end        : 根文件系统区域
--------------------------------
```

注意：请按照上面参数进行分区，否则我们提供 linux image 将不能正常启动！

vivi 一开始是利用 SJF2440 烧到 NANDflash 中的，上电后，VIVI 把自己从 NAND FLASH 拷贝到 SDRAM 运行。使用 bon 命令分区后，NAND FLASH 里面内容将全部擦除，包括 VIVI。(vivi 现在在 SDRAM 中运行，所以还可以看到)，因此，在进行 bon 分区后，不能按复位键或者关闭开发板电源，否则需要再一次使用 sjf2440 将 vivi 烧写到 NAND 中。

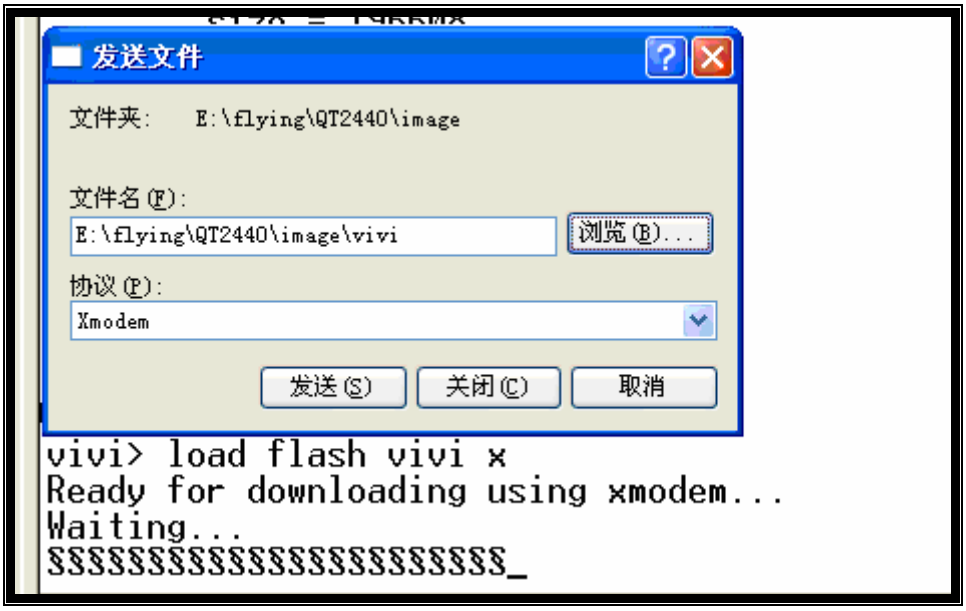
在超级终端中，vivi 下执行

```
============================
vivi>load flash vivi x
============================
```

输入上述命令之后，开发板便开始等待我们发送 vivi 映像文件，从超级终端菜单里选择"传送"下的"发送文件"选项，或者点击图标 ，出来对话框之后，在文件浏览栏选择 vivi 映像文件(在光盘\image 目录下)；此外，还要选择 xmodem 这个传输协议，点击“发送” 便可以看到数据传输的过程。

如下图所示：

选择待发送文件的界面



发送文件界面：

下面便可以来烧写内核 Image 了

在超级终端中，vivi 下执行

```
============================
vivi>load   flash   kernel   x
============================
```

与上一步类似，传送光盘\image 目录的"zImage"文件。该文件约 600K，传送时间较长，需要几分钟。

三、安装根文件系统

刚才只是烧写了 linux 的内核，如果需要一个可以真正跑起来的 linux，还得安装 linux 的根文件系统，因为，嵌入式系统的文件、应用程序都集中在这个部分里面。

在超级终端中，vivi 下执行

```
============================
vivi>     load flash root x
============================
```

与上面的步骤基本相同，发送 images 目录中的 "root.cramfs"映像文件。该文件约 1200K，传送时间较长，估计需要 5 分钟。

至此，linux 系统已经安装完毕，引导系统

```
============================
vivi>     boot
============================
```

这时您会看到串口输出大量的信息——这意味着，您亲手装的 linux 已经正常运行起来了。

使用指南二：如何在 Linux 下配置开发环境

该向导帮助用户安装 ARM 下 linux 开发必须的工具和代码，这是我们编译内核、vivi、应用程序的前提。以下工作需要在 linux 环境进行，请先安装 linux 操作系统。

一、拷贝文件包

以超级用户的身份登陆，并添加一个用户名为 test 的新用户；将软件包中的内容放到 "/home/test"目录中。其中有：

cross-2.95.3.tar.bz2	工具链
s3c2440_kernel2.4.18_rel.tar.bz2	核心代码
s3c2440_vivi_r1.0.tar.bz2	Vivi 代码
ztelnet-0.9.1-7mz.i386.rpm	PC 端软件，用网线与目标板 linux 传输文件

可以在光盘\LINUX\Linux 安装文件 目录下找到以上文件

二、安装 ToolChain

转到"/home/test"目录，按照下面的命令来安装：

```
=====================================================
[root@localhost test] # mkdir -p /usr/local/arm
[root@localhost test] # tar jxvf cross-2.95.3.tar.bz2
=====================================================
```

上面的命令将在"/test/"目录下 产生一个 "2.95.3" 文件夹，将此文件夹拷贝到 "/usr/local/arm/"目录下。

执行：

```
=====================================================
[root@localhost test]# mv 2.95.3 /usr/local/arm
[root@localhost test]# export PATH=$PATH:/usr/local/arm/2.95.3/bin
=====================================================
```

工具链（tool chain object files）如编译器、装载器等将被安装到"/usr/local/arm/2.95.3/bin"目录中

三、建立映像文件夹

为了后面更方便的工作，我们最好将编译好的映像文件集中到"/image"目录下。在"/home/test"目录下建立一个"/image"文件夹，将编译好的映像拷贝到该处

执行：

```
=======================
mkdir /home/test/image
=======================
```

建立好文件夹，不要忘了将上一个向导中用到的在光盘 image 目录中的 vivi、zImage 和 root.cramfs 拷贝到该目录下，方便以后使用。

四、安装 ztelnet

执行下面的命令来安装 RPM 包

```
==============================================
[root@localhost test]# rpm -i ztelnet-0.9.1-7mz.i386.rpm
==============================================
```

至此，基本的 ARM linux 开发环境建立完成，下一个向导将帮助您为目标板编译程序，并将之送到目标板上运行。

使用指南三：如何为目标板上的 Linux 编译程序

该向导帮助用户用 ARM 的编译器来编译两个程序，前一个向导（配置 linux 下开发环境）的完成是本向导的前提。

第一个程序便是"hello,world！"其运行的结果便是在控制台上打印"Hello,world！"

第二个程序是"imagewrite"这是一个 linux 系统下的 NANDflash 烧写程序，我们用过的烧写 NANDflash 的工具有两个：一个是"sjf2440 它是利用硬件 JTAG 下载的，因为速度慢，只是用于烧写 vivi；另一个便是 vivi，它可以分区也可以烧写整个操作系统，它使用串口下载，并且不支持大文件烧写，比如以后要烧写的 root_qtopia_2440.cramfs 这个文件，只有通过 imagewrite 烧写，无法使用 VIVI 下的 load flash root x 的方法进行；以上两者的速度都是很慢的。现在介绍的"imagewrite"也是一个 NANDflash 分区烧写工具，在基本的 linux 系统安装完毕后，我们便可以使用 ztelnet 通过网络来快速下载更大的文件系统映像，然后用"imagewrite"这个工具可以更方便的烧写。注意，这个工具是运行在目标板上 linux 下的一个应用程序。

一、拷贝源代码

先建立一个空文件夹

执行：

```
==========================================
[root@localhost test]# mkdir /home/test/app
[root@localhost test]# cd app
==========================================
```

将光盘\app 目录下的文件拷贝到刚才建立的"/home/test/app"文件夹中。

二、编译 imagewrite

执行下面的命令来生成 imagewrite 工具

```
=================================================================
[root@localhost app]# /usr/local/arm/2.95.3/bin/arm-linux-gcc -o imagewrite imagewrite.c
=================================================================
```

然后，将 imagewrite 工具拷贝到"/home/test/image"目录下

```
=======================================================
[root@localhost app]# cp imagewrite /home/test/image
=======================================================
```

注：如果不想自己编译改文件，可以直接使用光盘 image 目录下已经编译好的 imagewrite 文件

三、编译 hello,world!

执行下面的命令来编译 hello.c，并将目标文件拷贝到"/home/test/image"目录下

```
============================================================
[root@localhost app]# /usr/local/arm/2.95.3/bin/arm-linux-gcc -o hello hello.c
[root@localhost app]# cp hello /home/test/image
============================================================
```

至此，您已经将两个 C 语言文件编译成两个可执行文件，不过，这里的可执行只是针对开发板来说的——在 PC 机上是不可以运行的，您需要将其送到开发板上运行，如何做呢？请看下面的向导。

使用指南四：如何向目标板上的传输文件

该向导帮助用户使用 ztelnet 向目标板传送文件，通过网线下载前面提到的两个应用程序，最终我们让这两个程序在目标板上运行起来。

实践本向导的前提是 linux 下已经配置好的开发环境及两个应用程序映像。

在 linux 下打开一个终端（和 Windows 下的 DOS 界面类似）

执行：

```
==================================
[root@localhost root]# minicom -s
==================================
```

这将会打开一个 minicom 的配置窗口，配置细节中要求的关键参数如下：

```
==================================
Serial Device              :/dev/ttyS0
....
Bps/Par/Bits       :115200 8N1
Hardware Flow Control      :NO
Software Flow Control :NO
==================================
```

最后不要忘了选择：“Save Setup as df1”，将其保存为缺省设置

minicom 和 windows 下的超级终端很类似，在 linux 中我们使用它来代替超级中断。

配置完毕后，便可以运行 minicom，开发板加电，minicom 中会得到开发板上 linux 的控制台。

首先，我们要为目标板上的 Linux 配置 IP 地址，执行：

```
==================================
（QT2440 目标板侧）
# ifconfig    eth0    192.168.0.8
# inetd                 //非常重要，必须执行改指令
# ifconfig    eth0
==================================
```

第二个"ifconfig eth0"命令后会有一串的信息，可以从中看出，目标板的 IP 地址被定为"192.168.0.8"。现在，在主机侧，我们再打开一个终端，我们也需要给主机配置 IP 地址（如果你的 PC 机的 IP 地址在同一网段内则不必再设）

```
==================================================
（主机侧）
[root@localhost root]# ifconfig    down
[root@localhost root]# ifconfig    eth0    192.168.0.2 up
==================================================
```

这样主机和目标板便可以“看”到对方了，用户可以使用 PING 工具来测试是否连通，若否，请检查网线；检查主机和目标板的 IP 地址是否修改成功。当连通之后，我们便开始传输文件。

现在，主机桌面上应该有两个窗口，一个是连接目标板的 minicom，一个是主机自己的终端，我们执行：

==========
(在 QT2440 目标板终端：minicom)
#cd /tmp （注：目标板上 tmp 目录是 RAM DISK 是可读写的，同时掉电后文件将丢失）
==========

==========
（在主机终端窗口）
[root@localhost root]# cd /home/test/image （必须先进入这个目录再使用 ztelnet 连接目标板，因为下面要用到的 sz 命令后面的文件所在目录就是当前目录）
[root@localhost root]# ztelnet 192.168.0.8
（none）login : root
==========

上面的指令是使用 PC 机端的 ztelnet 工具登陆到目标板上，并用 root 用户名登陆。
下面的内容便是传输文件的过程了，需小心操作（可能还要快速），初学者失败的可能性很大

==========
（主机）
 # cd /tmp （注：必须先到 tmp 目录下，只有这里是可写的）
 # rz （ 回车后可以看到一些数据乱码，不要理会，按 ' Ctrl+]' Ctrl 加右方括号键回到 shell）
ztelnet> sz hello （回到 shell 界面，用 sz 命令传送 hello 文件，请确保该文件在/home/test/image 目录）
==========

这样，hello 文件被传到目标板上的"/tmp"目录下面，同样手法传送 imagewrite。（注意，这个步骤稍微有些麻烦，请用户多试验几次，掌握使用 ztelnet 的要领；此外，如果不成功，首先看是不是没有登陆上，只要你输入'root'完成后，连接就已经建立起来了；其次检查登陆后是不是 cd 到 tmp 目录下；再次，还要检查登陆前的当前目录是不是存在两个待传输的文件，这里假设的是 /home/test/image）

传输成功后，可在 minicom 中的控制台中输入"ls"命令来看到新的文件。

现在可以运行程序了，执行

==========
(目标板终端)
#cd /tmp
#./hello (然后，你应该可以看到 hello,world!输出)
==========

如果遇到提示文件权限不够的情况，请使用“chmod 777 文件名”来修改文件权限。

此外，传输过来的 imagewrite 程序是用来给 NANDflash 分区烧写用的，我们将在下一个向导中介绍如何使用这个程序。

使用指南五：如何使用 imagewrite 工具

检查/home/test/image 目录下是否有 VIVI zImage imagewrite 等文件，并把光盘 image 目录下的 root_qtopia_2440.cramfs 文件也拷贝到该目录，方便后面的操作。

下面需要用户按照前面提供的方法将软件包中的四个文件送到目标板上"/tmp"目录下。

传输完毕后。

在 minicom 中，执行：

```
==========================================
(目标板终端)
#cd /tmp
#./imagewrite /dev/mtd/0 -part 0 192k 2m
(...等候分区完成...)
(执行完这个命令后，NAND FLASH 被擦空，在没有执行下面三个操作之前，不要断电或者复位开发板）
#./imagewrite /dev/mtd/0 vivi:0
#./imagewrite /dev/mtd/0 zImage:192k
#./imagewrite /dev/mtd/0 root_qtopia_2440.cramfs:2m
（这样你便重新烧写了 NandFlash 上的印象文件）
==========================================
```

其实，上面的过程不需要如此复杂，如果需要换一个文件系统，不需要重新分区，只需要执行最后一句“**./imagewrite /dev/mtd/0 root_qtopia_2440.cramfs:2m**”只烧写文件系统即可，如果执行了分区，则必须将 VIVI zImage root 全部重新烧写。

文件体积过大时，使用 ztelnet 来下载，使用 imagewrite 来烧写，会帮助你省很多的时间——但是，前提是你要先有一个可以提供 TCP\IP 服务的 linux 系统。

烧写完毕后，按板上的 RST 键重新启动，一个新的系统便出现了（主要是文件系统不同，这一次烧进去的文件系统有 40M，包含了更多的程序与特性）。

系统支持根据出厂需要，可能设置为 USB 鼠标模式或者触摸屏模式。因为两个模式不能同时有效，所以请确认是使用鼠标还是触摸屏。如果不是你想要的模式，请按照《QT2440 从这里开始》中的介绍，重新制作烧写 **root_qtopia_2440.cramfs** 这个 root 文件）。

系统有两个 USB HOST，P7 是专用 HOST，只要空着就可以使用。P3 的 HOST 端口和 P8 的 device 端口共用，如果需要使用 P3 作为 HOST，把 J11,J12 的跳线差到标有 HOST 的位置才可以使用 P3 的 HOST 功能。

把 U 盘插入 HOST 端口，进入串口终端（minicom 或者 windows 的超级终端），如果没有看到 shell 提示符，多按几次回车就可以出现。在 tmp 目录下建立一个 flash 文件夹，然后使用“mount /dev/sda1 /tmp/flash”挂载 U 盘，进入/tmp/flash 目录就可以查看 U 盘内容了。操作过程如下：

```
hub.c: USB new device connect on bus1/1, assigned device number 2
scsi0 : SCSI emulation for USB Mass Storage devices
Vendor: Generic   Model: USB Flash Drive   Rev: %z!Y
Type:   Direct-Access                      ANSI SCSI revision: 02
Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
SCSI device sda: 254208 512-byte hdwr sectors (130 MB)
sda: Write Protect is off
Partition check:
/dev/scsi/host0/bus0/target0/lun0:<7>usb-storage: queuecommand() called
```

```
 p1
```
（以上是插入 U 盘后系统的提示信息）
```
[root@(none) /]# mkdir /tmp/flash
[root@(none) /]# mount /dev/sda1 /tmp/flash
cramfs: wrong magic
modprobe: modprobe: Can't open dependencies file /lib/modules/2.4.18-rmk7-pxa1/m
odules.dep (No such file or directory)
modprobe: modprobe: Can't open dependencies file /lib/modules/2.4.18-rmk7-pxa1/m
odules.dep (No such file or directory)
modprobe: modprobe: Can't open dependencies file /lib/modules/2.4.18-rmk7-pxa1/m
odules.dep (No such file or directory)
modprobe: modprobe: Can't open dependencies file /lib/modules/2.4.18-rmk7-pxa1/m
odules.dep (No such file or directory)
```
（一些错误信息，无需理会）
```
[root@(none) /]# cd /tmp/flash
[root@(none) flash]# ls
```
（显示的就是 U 盘里面的内容了）

当然我们不能等到关闭系统电源后再把的 U 盘拔下来，运行下面的命令后，你就可以安全的把你的 U 盘把下来了 。

umount /tmp/flash

注意：在执行这条命令之前要退到根目录下。

当然，当你挂载好你的 U 盘后，你就可以通过图形界面来访问了，这样你就可以轻松的实现 MP3 和 MP4 的功能了。

使用指南六：如何构造自己的文件系统映像

按照前面的介绍，我们想要在目标板上运行程序，总要来一次上传，这很不方便——如果你的程序要每次开机时运行就不可以了；这里向您介绍如何将自己的应用程序放到文件系统映像里，烧到 NANDflash 中，开机之后，您就会发现：你的程序已经在文件系统里了。

前面用到的 cramfs 文件是一种压缩格式的文件系统包，若想要把自己的程序加进去，需要以下流程：

将 cramfs 文件包解压到其他目录-->将编译好的程序放到解压后的文件夹里面适当的目录-->将 cramfs 文件包压缩-->烧写到 NAND 中。

首先，需要建立一个新的文件夹“/home/test/root_dir”，将光盘\image 目录中的 root.cramfs 和 root_qtopia_2440.cramfs 文件拷贝到/home/test/image 目录下。

```
================================================
[root@localhost root]#mkdir /home/test/root_dir    ;创建目录
================================================
```

同时可以看出，文件系统包小的只有 1M，大的足有 40M；这是因为前者是经过削减的--还记得前面介绍装系统的过程么？利用 vivi 烧写 1M 大小的文件系统就要 5 分多钟，你能忍受烧写 40M 的东西所需要的时间么？提供小型的文件系统目的是帮助建立一个基本的运行环境，能够运行网络，从网络进行数据传输，为烧写大的文件系统提供跳板。

我们先来看一下 cramfs 文件到底是什么

```
================================================================
[root@localhost root]#cd /home/test/root_dir    ;进入目录
[root@localhost root_dir]#mkdir /home/test/mnt    ;构造挂载点
[root@localhost root_dir]#mount -t cramfs -o loop /home/test/image/root.cramfs    /home/test/mnt ;
要保证 root.cramfs 在你的/home/test/image/下面
================================================================
```

这样，原本开发板上的文件系统就被安装在 /home/test/mnt 目录下了，你可以进去看一下；如果有兴趣也可以将 root_qtopia_2440.cramfs 挂载上去看看里面的内容。

虽然我们可以看到里面的内容，但是修改却是不允许的，否则我们可以直接改动里面的内容了；也许聪明的你一下便想到了——我们可以先将挂载后的目录 COPY 出来，修改后再压缩回 cramfs 便可以了。这的确是个好办法，但是也是 BUG 的源头——在这个文件系统里面充满了各种链接文件，一旦拷贝出去，便很可能走样了。

因此，挂载出来文件系统中的内容只可以作为参考，不好完全的使用，我们需要先将这个目录打包成 tar 文件，然后把它解压到适当的目录，就可以对它进行修改了，修改完后，再将它制作成 cramfs 文件就可以了。

下面开始实际操作，目标是将前面编译的 hello 程序放到目标板/usr 目录下：

打包

==========

cd /home/test

tar cvf root.tar mnt

cp root.tar /home/test/root_dir

==========

解压，执行：

==========

cd /home/test/root_dir

[root@localhost root_dir]#tar xvf root.tar

==========

解压之后会出现/mnt 目录，将你的 hello 或 imagewrite 放到/home/test/root_dir/mnt/usr 目录下

下面压缩该目录使之成为 cramfs 文件，并放到/home/test/image 下面，其名为“myfs.cramfs”

==========

[root@localhost root_dir]#./mkcramfs /home/test/root_dir/mnt /home/test/image/myfs.cramfs

==========

这样我们便得到了 myfs.cramfs 这个文件系统印象，试试将其烧到目标板上的 root 分区，启动起来，然后你会看到/usr 目录下便会有你放进去的文件，敲入“./hello” 运行一下。

同样的道理，你也可以修改 root_qtopia_2440.cramfs，构造你自己的文件系统。

使用指南七：如何编译 vivi

如果您已经按前面的向导配置过 linux 下的开发环境，那么"s3c2440_vivi_r1.0.tar.bz2 "包应该已经在您的"/home/test"目录下了。

使用下面的命令来解压和配置：

```
============================================================
[root@localhost test]# tar    jxvf    s3c2440_vivi_r1.0.tar.bz2
[root@localhost test]# cd    s3c2440_vivi_r1.0
============================================================
```

编译，执行：

```
=======================================
[root@localhost s3c2440_vivi_r1.0]#make
=======================================
```

如果编译工作顺利的话，"vivi"的 Bin 文件将会生成在当前目录下，将其放到 image 目录下：

```
============================================================
[root@localhost s3c2440_vivi_r1.0]# cp vivi /home/test/image
============================================================
```

至此，编译完成

使用指南八：如何编译内核

如果您已经按前面的向导配置过 linux 下的开发环境，那么" s3c2440_kernel2.4.18_rel.tar.bz2 "包应该已经在您的"/home/test"目录下了。

内核源码压缩包的名字是"linux-2.4.18-rmk7-pxa1-mz4.tar.bz2"。执行下面的命令来解压 bz2 包

```
==============================================================
[root@localhost test]# tar    jxvf    s3c2440_kernel2.4.18_rel.tar.bz2
[root@localhost test]# cd    s3c2440_kernel2.4.18_rel
==============================================================
```

LCD 补丁程序：如果你使用的 LCD 不是标准配置的 3.5” TFT 液晶屏，请把光盘\Linux 安装文件目录下的 s3c2410fb.c 拷贝到 kernel/driver/video 目录下，覆盖原文件，否则 LCD 显示将不正常。

进入生成的"kernel"目录，执行"make menuconfig"命令来配置参数，缺省的参数是为开发板优化的，内核的缺省参数配置文件在"kernel"目录下。

执行下面的命令来生成内核映像

```
==========================================================
[root@localhost kernel]#make dep
[root@localhost kernel]#make zImage
==========================================================
```

映像文件产生在" kernel/arch/arm/boot"目录下，名称为"zImage"，将其拷贝到"image 目录中"

```
==========================================================
[root@localhost kernel]cd arch/arm/boot
[root@localhost boot]#cp zImage /home/test/image
==========================================================
```

注意，这样会把原来的 zImage 覆盖掉。如果需要更新 zImage 文件，需要使用 ztelnet 和 imagewrite 来更新 kernel，请回到前面章节了解。

至此，你已经知道如何编译内核了，在以后的 linux 开发中，如果涉及到驱动或其他什么较底层的应用，你会经常需要配置、编译内核的，这就需要你对 linux 系统有较深刻的理解了。

结束语

-+---+---++-+--+-++-+++--++--+++-+-++--+-++-+--+-+---+---++-+--+-++-+++--++--+++-+-++--+

到此为止，本开发手册的内容就要结束了；我们的努力只能帮助你快速的跨过 **linux** 开发这一道门槛；要想深入到 **linux** 的广阔世界，你的面前还有很长的路要走，需要你对 **linux** 系统本身有深刻的理解，这些只能靠你自己去学习，去研究了。

此外，我们的论坛上有着众多的用户和爱好者，大众的资源会给您带来意想不到的帮助；欢迎您访问“**bbs.edw.com.cn**”中“**ARM** 开发论坛”，和大家一起，提出您的问题、发表您的成果。

-+---+---++-+--+-++-+++--++--+++-+-++--+-++-+--+-+---+---++-+--+-++-+++--++--+++-+-++--+

此外，在使用本手册的过程中如果遇到什么问题，也请您千万不要迟疑，马上写信给我们，我们非常期望您的反馈。

support@armzone.com